# Towards Cyclic Implicit Complexity

Gianluca Curzi
University of Birmingham
g.curzi@bham.ac.uk

Anupam Das
University of Birmingham
A.Das@bham.ac.uk

**Introduction.** Formal proofs are traditionally seen as finite mathematical objects modelling logical or mathematical reasoning. Non-wellfounded (but finitely branching) proofs represent a generalization of the notion of formal proof to an infinitary setting. The proof theory of non-wellfounded proofs originates in the context of the modal $\mu$-calculus [NW96] and has been subject to growing interest in recent years.

In this framework, a special attention is devoted to *circular proofs*, i.e. those non-wellfounded proofs having only finitely many distinct sub-prooftrees. Circular proofs can be turned into finite structures called *cycle normal forms*, usually defined as finite trees with additional 'backpointers'. As circular proofs admit fallacious reasoning, a standard solution to prevent inconsistency is to introduce non-local *correctness criteria*, typically checked by Büchi automata on infinite words.

Circular proofs, and their corresponding cycle normal forms, have been employed to reason about modal $\mu$-calculus and fixed-point logics [DHL06], induction and coinduction [BS11], Kleene algebra [DP17], linear logic [BDS16], arithmetic [Das18], and continuous cut-elimination [Min78, FS13]. However, little is known about their complexity-theoretic aspects. The present paper aims at bridging the gap between circular proofs and Implicit Computational Complexity (ICC), a branch of computational complexity studying machine-free languages and calculi able to capture a given complexity class without relying on explicit resource bounds.

**A proof system for $\mathsf{B}$.** Our starting point is the Bellantoni and Cook's function algebra $\mathsf{B}$ capturing the polynomial time computable functions (**FPTIME**) in the spirit of ICC using *safe recursion*. Functions of $\mathsf{B}$ have shape $f(x_1,...,x_n;y_1,...,y_m)$, where the semicolon separates the *normal* arguments $x_1,...,x_n$ from the *safe* arguments $y_1,...,y_m$. The idea behind safe recursion is that only normal arguments can be used as recursive parameters, while recursive calls can only appear in safe position. This prevents recursive calls to become recursive parameters of other previously defined functions. $\mathsf{B}$ can be alternatively designed as a proof-system deriving sequents with shape

$$\overbrace{\Box N,\dots,\Box N}^{n},\overbrace{N\dots,N}^{m}\Rightarrow N$$

where $N$ is the ground type for natural numbers and $\Box N$ is its modal version. In this system, safe recursion is introduced by a specific inference rule $\mathsf{srec}$. Intuitively, a proof of the above sequent represents a Bellantoni and Cook's function with $n$ normal arguments and $m$ safe arguments.

**$\mathsf{NCB}$ and nesting.** Starting from $\mathsf{B}$, we obtain the circular proof system $\mathsf{NCB}$ in three steps:

- we consider the non-wellfounded proofs generated by the rules of the subsystem $\mathsf{B}^- := \mathsf{B}\setminus\mathsf{srec}$, which are able to subsume various forms of recursion;
- we consider the circular proofs of $\mathsf{B}^-$ that satisfy a termination criterion, so that only total computable functions are representable;

To appear in EPTCS.

- we introduce a 'safety' criterion in order to prevent non-safe recursion schemes to be represented in this infinitary setting.

Despite NCB having only ground types, it is able to define safe recursion schemes that nest the recursive calls, a property that typically arises in higher-order recursion. As an example, NCB is able to represent the exponential function $\mathsf{exp}(x) = \lambda y.2^x + y$ defined as follows:

$$\begin{array}{rcl} \mathsf{exp}(0) & = & \lambda y.y+1 \\ \mathsf{exp}(x+1) & = & \lambda y.\mathsf{exp}(x)(\mathsf{exp}(x)(y)) \end{array}$$

This is in fact a peculiar feature of circular proofs which has been extensively studied by Das in [Das21], who has shown that the number-theoretic functions definable by type level $n$ proofs of a circular version of system T are exactly those ones definable by type level $n+1$ proofs of T. To make this point more apparent, consider the following higher-order recursion operator:

$$\mathsf{rec}_A : \Box N \to (\Box N \to A \to A) \to A \to A$$

with $A = N \to N$, and $f(x) = \mathsf{rec}_A(x,h,g)$ is defined as $f(0) = g$ and $f(x+1) = h(x, f(x))$ for $x > 0$. By setting $g := \lambda y : N.y+1$ and $h := \lambda x : \Box N.\lambda u : N \to N.(\lambda y : N.(u(u(y))))$ we can easily check that $\mathsf{exp}(x;y) = \mathsf{rec}_A(x,h,g)(y)$. The fact that higher-order safe recursion can be used to define the class of the elementary time functions (**FELEMENTARY**) has already been noticed by Hofmann in [Hof97], where the type system SLR (Safe Linear Recursion) is introduced to lift B to a higher-order setting. The system uses a variant of the operator $\mathsf{rec}_A$ with a linearity restriction preventing the recursive calls to be duplicated, and hence nested.

CB **and characterizations.** Following [Hof97], we impose a linearity criterion to rule out the nesting of recursive calls by controlling the interplay between loops and the cut rule. The resulting circular proof system is called CB. Intuitively, CB can be seen as a circular version of B, while NCB generalizes CB allowing nested versions of the safe recursion scheme. The main results of our paper are the following:

**Theorem.**

- *A function is representable in* NCB *iff it is in* **FELEMENTARY**.
- *A function is representable in* CB *iff it is in* **FPTIME**.

Completeness is easily achieved by showing how to represent a given class of functions in the circular proof system. By contrast, soundness is subtler, as it relies on a translation of circular proofs, which are essentially coinductive objects, into the functions of an inductively defined function algebra. In particular, in order to define the nesting of recursive calls, the function algebra for NCB crucially requires the introduction of 'auxiliary functions' (or oracles).

**Conclusions.** The widespread approach to ICC is based on the introduction of inductively defined languages or calculi endowed with recursion mechanisms whose strength is carefully calibrated in order to increase in complexity while not overstepping a given bound on computation. The circular proof systems CB and NCB pave the way to a radically different, top-down approach, where coinductive reasoning plays a central role. Circular proofs are able to subsume various forms of recursion, and in some cases are even able to express non-terminating computation. Therefore, the characterization results are achieved by imposing global proof-theoretical conditions weakening the computational content of the system.

As a future direction, we are planning to design higher-order versions of NCB and CB to reformulate the above results into a more general setting.

# References


[BDS16] David Baelde, Amina Doumane, and Alexis Saurin. Infinitary proof theory: the multiplicative additive case. 2016.

[BS11] James Brotherston and Alex Simpson. Sequent calculi for induction and infinite descent. *Journal of Logic and Computation*, 21(6):1177–1216, 2011.

[Das18] Anupam Das. On the logical complexity of cyclic arithmetic. *arXiv preprint arXiv:1807.10248*, 2018.

[Das21] Anupam Das. On the logical strength of confluence and normalisation for cyclic proofs. In Naoki Kobayashi, editor, *6th International Conference on Formal Structures for Computation and Deduction, FSCD 2021, July 17-24, 2021, Buenos Aires, Argentina (Virtual Conference)*, volume 195 of *LIPIcs*, pages 29:1–29:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[DHL06] Christian Dax, Martin Hofmann, and Martin Lange. A proof system for the linear time $\mu$-calculus. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 273–284. Springer, 2006.

[DP17] Anupam Das and Damien Pous. A cut-free cyclic proof system for kleene algebra. In *International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 261–277. Springer, 2017.

[FS13] Jérôme Fortier and Luigi Santocanale. Cuts for circular proofs: semantics and cut-elimination. In *Computer Science Logic 2013 (CSL 2013)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[Hof97] Martin Hofmann. A mixed modal/linear lambda calculus with applications to bellantoni-cook safe recursion. In Mogens Nielsen and Wolfgang Thomas, editors, *Computer Science Logic, 11th International Workshop, CSL '97, Annual Conference of the EACSL, Aarhus, Denmark, August 23-29, 1997, Selected Papers*, volume 1414 of *Lecture Notes in Computer Science*, pages 275–294. Springer, 1997.

[Min78] Grigori E Mints. Finite investigations of transfinite derivations. *Journal of Soviet Mathematics*, 10(4):548–596, 1978.

[NW96] Damian Niwiński and Igor Walukiewicz. Games for the $\mu$-calculus. *Theoretical Computer Science*, 163(1-2):99–116, 1996.